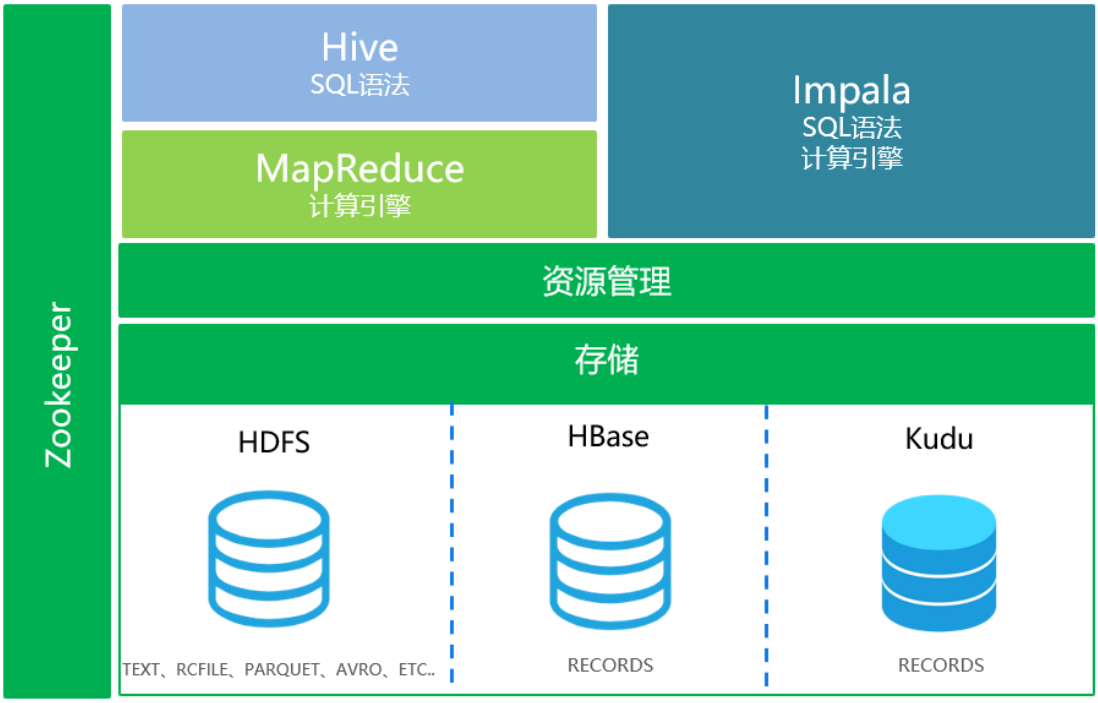
1. 集成Impala

1.1. 各组件之间的关系

Kudu 整合 Impala依赖很多组件，这里有一张组件关系图如下：



通过上图我们分析结论如下：

```
impala依赖Hive
Hive依赖Hadoop
Hive依赖MySQL存储元数据
Hadoop依赖Zookeeper
基本都离不开JDK
```

因此，相关组件我们需要先安装。

1.2. 环境准备

请自行准备3台服务器：bigdata02, bigdata03, bigdata04

1.3. 部署规划

具体规划如下：

应用	bigdata02	bigdata03	bigdata04
kudu-master	kudu-master	kudu-master	kudu-master
kudu-tserver	udu-tserver	udu-tserver	udu-tserver
ntpd	ntpd	ntpd	ntpd
JDK-1.8	JDK-1.8	JDK-1.8	JDK-1.8
MySQL-5.7	MySQL-5.7		
ZooKeeper	ZooKeeper	ZooKeeper	ZooKeeper
Hadoop	NameNode ResourceManager SecondaryNamenode HistroyServer Hadoop Client	NodeManager DataNode	NodeManager DataNode
Hive	hive-metastore hive- server2 hive		
Impala	impala-state-store impala-catalog	impala-server	impala-server

1.4. 安装JDK

见 JDK 安装文档

1.5. 安装MySQL

见MySQL安装文档

1.6. 安装Zookeeper

见zookeeper安装文档

1.7. 安装Hadoop

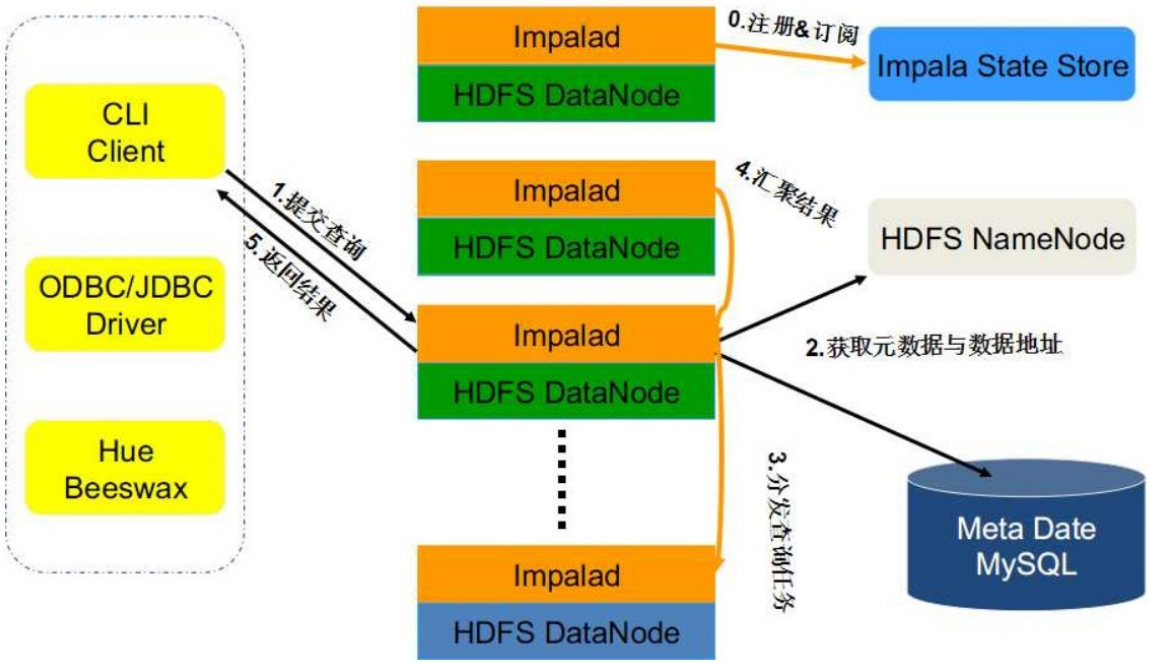
见 Hadoop 安装文档

1.8. 安装 Hive

见 Hive 安装文档

1.9. 安装 Impala

1.9.1. Impala架构



服务	作用
Catalog	Impala的元信息仓库, 但是不同的是这个Catalog强依赖 Hive的 MetaStore, 会从Hive处获取元信息
StateStore	Impala 的协调节点, 负责异常恢复
ImpalaServer	Impala 是MPP架构, 这是Impala处理数据的组件, 会读取HDFS, 所以一般和DataNode部署在一起, 提升性能, 每个DataNode配一个ImpalaServer

1.9.2. 部署 Impala

bigdata02 上部署StateStore、Catalog

```
sudo yum install -y impala impala-state-store impala-catalog impala-shell
```

bigdata03 和 bigdata04 上部署ImpalaServer

```
sudo yum install -y impala impala-server
```

bigdata03 和 bigdata04 上部署 mysql 的 jdbc 驱动包

```
sudo yum install -y mysql-connector-java
sudo ln -s /usr/share/java/mysql-connector-java.jar /usr/lib/hive/lib/mysql-
connector-java.jar
```

1.9.3. 配置 Impala

在所有节点执行如下操作：链接 hadoop 和 hive 配置文件

Impala 需要访问 hadoop 和 hive，因此需要以软连接的方式把配置文件放到 Impala 的配置目录下

```
sudo ln -s /etc/hadoop/conf/core-site.xml /etc/impala/conf/core-site.xml
sudo ln -s /etc/hadoop/conf/hdfs-site.xml /etc/impala/conf/hdfs-site.xml
sudo ln -s /etc/hive/conf/hive-site.xml /etc/impala/conf/hive-site.xml
```

配置 Impala 的环境变量：DH 把各个组件的环境变量文件放在 /etc/default 下：

```
sudo vi /etc/default/impala
```

配置内容如下：

```
IMPALA_CATALOG_SERVICE_HOST=bigdata02
IMPALA_STATE_STORE_HOST=bigdata02
```

其余配置保持默认即可。

1.9.4. 启动 Impala

在 bigdata02 启动 state-store 和 catalog：

```
#启动有先后顺序的
sudo service impala-state-store start
sudo service impala-catalog start
```

在 bigdata03 和 bigdata04 启动 impala-server：

```
sudo service impala-server start
```

1.9.5. 验证 Impala

访问 statestore 的 web ui：http://bigdata02:25010/

访问 catalog 的 web ui：http://bigdata02:25020/

访问 impala-server 的 web ui：http://bigdata03:25000 或者 http://bigdata04:25000

```
sudo vi /etc/default/impala
```

修改内容如下：

```
IMPALA_CATALOG_SERVICE_HOST=bigdata02
IMPALA_STATE_STORE_HOST=bigdata02
IMPALA_STATE_STORE_PORT=24000
```

```
IMPALA_BACKEND_PORT=22000
IMPALA_LOG_DIR=/var/log/impala
IMPALA_CATALOG_ARGS=" -log_dir=${IMPALA_LOG_DIR} "
IMPALA_STATE_STORE_ARGS=" -log_dir=${IMPALA_LOG_DIR} -
state_store_port=${IMPALA_STATE_STORE_PORT}"
IMPALA_SERVER_ARGS=" \
    -log_dir=${IMPALA_LOG_DIR} \
    -catalog_service_host=${IMPALA_CATALOG_SERVICE_HOST} \
    -state_store_port=${IMPALA_STATE_STORE_PORT} \
    -use_statestore true \
    -state_store_host=${IMPALA_STATE_STORE_HOST} \
    -be_port=${IMPALA_BACKEND_PORT}"
```

1.10. Kudu+Impala整合

1.10.1. 为什么整合Kudu+Impala

Kudu 作为高性能的分布式存储同时兼具 HDFS 和 HBase 的能力确实能够解决很多业务问题，但是 Kudu 没有 SQL 语法支持限制它的使用门槛，因此 Cloudra官方专门把 Impala 和 Kudu 做了整合，因此他们的分工是：

```
Kudu负责存储
Impala负责计算（用SQL语法分析存储在Kudu表里的数据）
```

Impala对外支持两种场景：

```
基于Impala替代Kudu API开发上层应用（可以但不推荐）
Impala作为中间层提供JDBC/ODBC跟上层BI或者其他框架整合（推荐）
```

1.10.2. 怎么整合Kudu+Impala

何为整合？让Impala认识并能操作Kudu中的表（内部表/外部表均可）：

存储	说明	特点
内部表	在Impala中创建表存储为Kudu格式，它会自动在Kudu中创建相应的表，并维护好映射关系	Kudu中的表跟Impala中的”同生同死”
外部表	Kudu中已经存在一张表，再映射到Impala作为外表	Impala中删除外表只会删除映射关系，不会删除Kudu那边的表

Kudu 跟 Impala 的整合非常简单，可总结为两点：

```
Kudu 这边什么配置都不用改，它就等着 Impala 来访问
Impala 那边有两种方式来访问 Kudu
```

方式一：每次在 Impala 中建内部表/外部表时指定 Kudu Master（不推荐）

```
CREATE EXTERNAL TABLE `users` STORED AS KUDU TBLPROPERTIES('kudu.table_name' =
'users', 'kudu.master_addresses' =
'bigdata02:7051,bigdata03:7051,bigdata04:7051')
```

方式二：在 Impala 的默认配置中指定 Kudu Master+内部表/外部表

在 /etc/default/impala 中指定：

```
--kudu_master_hosts=<master1>[:port],<master2>[:port],<master3>[:port]
```

1.10.3. 整合Kudu+Impala

1.10.3.1. 配置Kudu Master地址

在Impala中配置Kudu Master地址（所有节点）：

```
sudo vi /etc/default/impala
```

在IMPALA_SERVER_ARGS 下添加如下配置：

```
-kudu_master_hosts=bigdata02:7051,bigdata03:7051,bigdata04:7051
```

1.10.3.2. 重启所有 Impala 服务

bigdata02 重启 state-store 和 catalog：

```
sudo service impala-state-store restart
sudo service impala-catalog restart
```

bigdata03 和 bigdata04 上重启impala-server：

```
sudo service impala-server restart
```

1.10.4. Impala Shell 中操作 Kudu

1.10.4.1. 登录 Impala-shell

我们在 bigdata02 上安装了 impala-shell，因此在 bigdata02 上执行如下命令：

```
impala-shell -i bigdata02:21000
```

1.10.4.2. 表映射

Impala可以操作很多表：

```
1、Kudu表
2、HBase表
3、Hive表（各种存储格式：Text、ORC、Parquet等等）
```

本课程之讲解 Impala 怎么操作 Kudu 表。Impala 要想操作 Kudu 表，有两种方式：

外部表：所谓外部表是指Kudu那边已经建好表了，我们把Kudu表映射为一张Impal表即可，删除表时只删映射关系，Kudu那边的表还在。以前面章节创建的表students1为例，只需要在impala-shell中创建一个外部表即可：

```
CREATE EXTERNAL TABLE `students1` STORED AS KUDU TBLPROPERTIES('kudu.table_name'
= 'students1');
```

或者在指定数据库下创建外表：

```
CREATE DATABASE IF NOT EXISTS test;

CREATE EXTERNAL TABLE test.users STORED AS KUDU TBLPROPERTIES('kudu.table_name'
= 'users');
```

内部表：所谓内部表，它跟外部表正好相反，是指在Impala中创建一张表存储为Kudu格式，例如：

```
CREATE TABLE my_first_table(
  id BIGINT,
  name STRING,
  PRIMARY KEY(id)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU
TBLPROPERTIES ('kudu.num_tablet_replicas' = '1');
```

这时他会自动在 Kudu 中也创建一个表：注意：删除内部表，Kudu 中的表是会删除的。

1.10.4.3. 查询

查询就是 SQL 语法，大家可以自行尝试，例如：

```
select gender, count(*), max(height) ,min(height), avg(height) from students1
where age <= 19 and height > 180 group by gender;
```

1.10.4.4. DML

插入数据

```
#单行插入
INSERT INTO my_first_table VALUES (1, "zhangsan");
select * from my_first_table;
```

```
#多行插入
INSERT INTO my_first_table VALUES (2, "lisi"), (3, "wangwu"), (4, "zhaoliu");
select * from my_first_table;
```

```
CREATE TABLE test2(
  id BIGINT,
  name STRING,
  PRIMARY KEY(id)
)
PARTITION BY HASH PARTITIONS 6
STORED AS KUDU
TBLPROPERTIES ('kudu.num_tablet_replicas' = '1');

select * from test2;

#从其它表批量导入
INSERT INTO test2 SELECT * FROM my_first_table;
select * from my_first_table;
```

更新数据

```
#更新
UPDATE my_first_table SET name="张三" where id =1 ;
select * from my_first_table;
```

删除数据

```
delete from my_first_table where id =3;
select * from my_first_table;
```

1.10.4.5. 更改表属性

重命名impala内部表

```
ALTER TABLE my_first_table RENAME TO person;
show tables;
```

Kudu那边的表也跟着改名了

重命名impala外部表

```
ALTER TABLE students1 RENAME TO stus;
show tables;
```

Kudu那边的表名不会跟着改变（只是改了映射）

将外部表重新映射kudu表

```
ALTER TABLE external_table
SET TBLPROPERTIES('kudu.table_name' = 'xxx')
```

将内部表改为外部表

```
ALTER TABLE my_table SET TBLPROPERTIES('EXTERNAL' = 'TRUE');
```

1.11. 谓词下推

所谓谓词简单理解就是 SQL 的 where 字句中的条件判断，Impala 的原理就是读取 Kudu 表的数据然后进行计算，如果谓词能够下推到 Kudu 中去执行则返回给Impala 的数据将会很小，性能将大幅提升。目前：

```
支持下推的谓词：=, <=, <, >, >=, BETWEEN, IN
不支持下推的谓词： !=, LIKE, 或者Impala中的其他谓词
```